Opus Coarse Energy Predictor Notes

Jake Taylor (yupferris@gmail.com), 27 June 2021

I'm having trouble reconciling the coarse energy predictor implementation in the libopus source code and the 2D z-transform description in the paper[1].

I've simplified the source code (in unquant_coarse_energy in quant_bands.c in libopus 1.3.1[2]) to the following C-like pseudocode:

```
void unquant_coarse_energy(float *e, int bands) {
  float alpha = /* ... */;
  float beta = /* ... */;
  float p = 0.0f;
  for (int b = 0; b < bands; b++) {
    float q = /* read from bitstream */;
    e[i] = alpha * e[i] + p + q;
    p = p + q - beta * q;
  }
}
```

According to the paper, the 2D z-transform should be:

$$A(z_\ell, z_b) = (1 - \alpha z_\ell^{-1}) \cdot \frac{1 - z_b^{-1}}{1 - \beta z_b^{-1}}$$

First off, to state what I think is obvious: the domain of this filter should be a 2D "energy plane" with the ℓ-dimension representing frames and the b-dimension representing bands, and the range should be the prediction (actual band energy - $q[\ell, b]$, the residual). As a predictor, the filter must be causal. Finally, according to the code above, the energy is always 0 for $b < 0$ ($\ell < 0$, $b \geq bands$, and $\ell \geq frames$ are not specified nor useful).

Assuming this filter is separable, we first have the ℓ-dimension predictor:

$$A(z_\ell) = 1 - \alpha z_\ell^{-1}$$

At first, I thought this was clearly embodied by alpha $*$ `e[i]` above. However, the z-transform implies that it should actually be (1 − alpha) $*$ `e[i]`, so already we seem to be missing another `e[i]` term somewhere (not to mention alpha having the wrong sign).

The b-dimension predictor seems even more problematic:

$$A(z_b) = \frac{1 - z_b^{-1}}{1 - \beta z_b^{-1}}$$

This matches what's listed in the CELT blog post[3], and is equivalent to:

$$Y(z_b) = \frac{1 - z_b^{-1}}{1 - \beta z_b^{-1}} X(z_b)$$

The equivalent difference equation is:

$$y[b] = x[b] - x[b-1] + \beta y[b-1]$$

And substituting names from the C code, we should get something like:

$$prev[b] = q[b] - q[b-1] + \beta prev[b-1]$$

Now, it should be mentioned that I actually asked about this recently in the DSP stack exchange[4] (after first emailing Jean-Marc Valin directly, but I seem to have scared him off with another wall of text similar to this one), and a helpful user there was able to clarify many things. We actually arrived at the same difference equation in the end, even though we got there a bit of a different way (one which actually included both dimensions from the original 2D z-transform), which suggests that my analysis above is correct.

However, we still didn't figure out the last bit: reconciling it with the C code; it appears to differ. If I forget about the above and just read the C code, we should get:

$$prev[b] = prev[b-1] + q[b] - \beta q[b]$$

The equivalent z-transform for this difference equation would be:

$$A(z_b) = \frac{1 - \beta}{1 - z_b^{-1}}$$

This suggests that the actual predictor description might instead be:

$$A(z_\ell, z_b) = (1 - \alpha z_\ell^{-1}) \cdot \frac{1 - \beta}{1 - z_b^{-1}}$$

However, that still ignores the apparently-missing `e[i]` term from the ℓ-dimension.

So, what am I missing? One thing that I glossed over above that the first predictor dimenson (ℓ) appears to be applied to the band energy directly (as expected), whereas the second predictor dimension (b) appears to be applied to the residual q. Since q can be expressed in terms of the energy and the predictor, I tried several different interpretations and substitutions in various domains in order to describe a predictor in with the 2D "energy plane" as the domain and the prediction as the range, and got some crazy z-transforms that don't look correct; here's a few just for the curious:

$$A(z_b, z_\ell) = \frac{1 - \beta + \alpha z_\ell^{-1}(1 - z_b^{-1})}{\beta - z_b^{-1}}$$

$$A(z_b, z_\ell) = \frac{1 + \beta z_b^{-1} - \alpha z_\ell^{-1}(1 - z_b^{-1})}{(1 + \beta) z_b^{-1}}$$

So, at this point I'm kindof running in circles, and I think I may have done something wrong; at least I'd like to think that's a lot more likely than the paper/RFC/libopus code were out of sync somehow!

[1] https://arxiv.org/abs/1602.04845

[2] https://opus-codec.org/release/stable/2019/04/12/libopus-1_3_1.html

[3] https://jmvalin.dreamwidth.org/12000.html

[4] https://dsp.stackexchange.com/questions/75972/having-trouble-interpreting-z-transform-description-of-a-predictor-from-a-codec